REMARKS

Claims 1-15 remain pending in the case. Claims 1, 6 and 11 have been amended. No new matter has been added. For example, support for "resulting in an instrumented code space" can be found, among other places, at page 10 lines 32-37 of the instant application serial no. 10/017,342. Support for "resulting in an uninstrumented code space that is an uninstrumented version of said instrumented code space" can be found, among other places, at page 10 lines 13-17 of the instant application.

112 Rejections

Claims 1, 6 and 11

Claims 1, 6 and 11 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particular point out and distinctly claim the subject matter which applicants regard as the invention. Claims 1, 6 and 11 have been amended to recite "said instrumented code which is inaccessible due to being unmapped." Therefore, Applicants believe that this rejection has been addressed.

103 Rejections

Claims 1-15

Claims 1-15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Benitez et al. U.S. Patent No. 6,189,141 B1 (referred to hereinafter as "Benitez") in view of "Unix Programming Frequently Asked Questions –1. Process Control" (referred to hereinafter as "Unix"). Applicants have reviewed the cited references and the instant application and respectfully assert that the claimed embodiments of the present invention are neither taught nor suggested by Benitez nor Unix, alone or in combination.

Claim 1 recites,

> A computer-implemented method of reverting a process in an in-line instrumented state to an uninstrumented state, said method comprising:

receiving a child process having inherited an instrumented parent process' context including the parent's program text that may have been modified by instrumentation resulting in an instrumented code space;

modifying text segment portions selected from said child process resulting in an uninstrumented code space that is an uninstrumented version of said instrumented code space;

unmapping said instrumented code space such that said instrumented code space is inaccessible to said child process;

provided an instruction pointer resides in said instrumented code space, updating said instruction pointer to point to said uninstrumented code space; and

executing said child process and, provided said child process generates a fault by seeking to access an address in said instrumented code space which is inaccessible due to being unmapped, providing an address in said uninstrumented code space that corresponds to said address in said instrumented code.

Benitez teaches a method and a system for optiziming code by identifying and translating hot blocks. Benitez also teaches a method and system for reducing unnecessary translations and optimizations to increase the speed of executing code. This is accomplished by identifying traces within the original code and determining the frequency with which those traces are executed or emulated. If the traces are executed frequently, the traces are "hot traces" and are candidates for translation/optimization. However, if the traces are executed infrequently, the traces are "cold traces" and are not candidates for translation/optimization. The frequency with which the traces are executed is tracked and compared to a threshold to determine which traces are "hot" and which traces are "cold."

As is well known in the art, a jump instruction may cause execution to jump to a second instruction by specifying the target address of the second instruction. Assume for the sake of illustrating Benitez that at the beginning of executing a second portion of original instructions includes a jump instruction that causes execution to jump to the target address of another instruction in a first portion of original instructions. Also assume for the sake of illustrating Benitez that these two portions of original instructions are identified as hot blocks

and are translated into respective first and second translated hot blocks. The target address has changed due to the translations and therefore the target address that the jump instruction would jump to would need to be changed to correctly identify the corresponding target address in the first translated hot block. Benitez discusses using a backpatcher at Col. 3 lines 18-24 so that the jump instruction in the second translated hot block correctly causes execution to jump to the new target address in the first translated hot block.

Now assume for the sake of illustrating Benitez that the first translated hot block is identified as "cold" and therefore the first portion of original code will be executed instead of the first translated hot block. In this case, the target address specified by the jump instruction in the second translated hot block will need to be changed back. Benitez discuses using the backpatcher at Col. 29 lines 19-24 to correct the target address specified by the jump instruction.

Applicants respectfully agree with the Office Action's statement that Benitez does not teach or suggest, "receiving a child process having inherited an instrumented parent process' context including the parent's program text that may have been modified by instrumentation resulting in an instrumented code space." The Office Action states, "Further, Benitez does not explicitly disclose receiving a child process having inherited an instrumented parent process' context but does disclose receiving new processes (col. 23, lines 19-20 'creates a record in table 222... if a record does not already exist')." However, Benitez states at Col. 23 lines 19-20, "Arc designator 520 creates a record in table 222 for each such target block, if a record does not already exist" (emphasis added). Benitez's target block isn't a process. Benitez's target block is a sequence of instructions that may potentially be identified as a hot block.

Claim 1 recites, "modifying text segment portions selected from said child process resulting in an uninstrumented code space that is an uninstrumented version of said instrumented code space." In contrast, Benitez's original

instructions appear to always be present and the instrumented hot blocks are instrumented versions of portions of the original instructions. For example, refer to Col. 14 lines 44-46 which states, "...original instructions generally are emulated if they are not identified as part of a hot block, and therefore translated." Then referring to Col. 9 lines 51-56, the translated hot blocks are "removed" from memory when they are identified as being "cold."

It appears that the Office Action is asserting that Benitez's translated hot blocks are analogous to Claim 1's "instrumented code space." Since Benitez teaches removing translated hot blocks from memory, Benitez cannot teach "provided an instruction pointer resides in said instrumented code space, updating said instruction pointer to point to said uninstrumented code space," as Claim 1 recites.

Applicants respectfully agree with the Office Action's admission that Benitez does not disclose "executing said child process and, generates a fault by seeking to access an address to instrumented code space..." The Office Action goes on to assert,

> However Benitez does teach that control should be returned to fetcher 430 when (col. 30, 'If cold trace detector and remover 1220 had not been invoked, ... time may be spent returning control to instruction fetcher 430'), and It would have been obvious to a person of ordinary skill in the art at the time of the invention to raise a fault (col. 11, lines 28-38 'an error condition has been detected') in this instance as a means of returning control to the uninstrumented code (col. 30, 'returning control to instruction fetcher 430').

Applicants have reviewed all of Benitez and have not found anything in Benitez to teach or suggest, "executing said child process and, generates a fault by seeking to access an address to said instrumented code space..." Further, the statements that the Office Action assert are in Col. 30 were not found. Applicants respectfully request that the next Office Action specify the column and line numbers when citing a portion of a reference.

Regarding the Office Action's statement, "... in this instance as a means of returning control to the uninstrumented code," Applicants respectfully point out that Claim 1 does not recite, "means of returning control to the uninstrumented code." Claim 1 does recite "executing said child process and, provided said child process generates a fault by seeking to access an address in said instrumented code space which is inaccessible due to being unmapped, providing an address in said uninstrumented code space that corresponds to said address in said instrumented code."

Applicants respectfully agree with the Office Action that the Unix reference does not teach or suggest,

> modifying text segment portions selected from said child process resulting in an uninstrumented code space that is an uninstrumented version of said instrumented code space; unmapping said instrumented code space such that said instrumented code space is inaccessible to said child process; provided an instruction pointer resides in said instrumented code space, updating said instruction pointer to point to said uninstrumented code space; and ... provided said child process generates a fault by seeking to access an address in said instrumented code space which is inaccessible due to being unmapped, providing an address in said uninstrumented code space that corresponds to said address in said instrumented code.

The Office Action asserts that the Unix reference teaches "a child process having ..." However, clearly the Unix reference does not teach "receiving a child process..." (emphasis added).

The Office Action goes on to state, "The fork() function is used to create a new process from an existing process')." However, this still does not reach the limitation of "a child process having inherited an instrumented parent process' context including the parent's program text that may have been modified by instrumentation resulting in an instrumented code space."

The Office Action goes on to state, "It would have been obvious to a person of ordinary skill in the art at the time of the invention to submit any newly

created child process to Benitez' 'Cold Block Remover' (col. 27, lines 49-51) because 'A hot trace is a trace through which control... has passed more than a predetermined number of times (col. 2, line 41-44)." However, Applicants respectfully point out, assuming for the sake of argument (these are not admissions on the part of Applicants) that even if the Unix reference teaches a child process and even if the Unix reference's child process were submitted to Benitez' Cold Block Remover this still would not reach, among other things, "modifying text segment portions selected from said child process resulting in an uninstrumented code space that is an uninstrumented version of said instrumented code space...provided an instruction pointer resides in said instrumented code space, updating said instruction pointer to point to said uninstrumented code space," as recited by Claim 1 for reasons already discussed in the context of Benitez.

Therefore, the Unix reference does not remedy the deficiencies in Benitez for at least the reasons that neither reference teaches or suggests "modifying text segment portions selected from said child process resulting in an uninstrumented code space that is an uninstrumented version of said instrumented code space...provided an instruction pointer resides in said instrumented code space, updating said instruction pointer to point to said uninstrumented code space," as recited by Claim 1.

Independent Claims 6 and 11 should be allowable for similar reasons that Claim 1 should be allowable. Claims 2-5 depend from Claim 1, Claims 7-10 depend from Claim 6, and Claims 12-15 depend from Claim 11. Therefore, the dependent claims include all of the limitations of their respective independent claims. Further, the dependent claims include additional limitations which further make them patentable. Therefore, the dependent claims should be patentable for at least the reasons that their respective independent claims should be patentable.

CONCLUSION

In light of the above remarks, Applicant respectfully requests reconsideration of the rejected Claims 1-15.

Based on the argument presented above, Applicant respectfully asserts that Claims 1 through 15 overcome the rejections of record and, therefore, allowance of these Claims is respectfully solicited.

The Examiner is invited to contact Applicants' undersigned representative if the Examiner believes such action would expedite resolution of the present Application.

Respectfully submitted,

WAGNER, MURABITO & HAO LLP

Date: 07/06/06

John P. Wagner Jr.
Reg. No. 35,398

Two North Market Street
Third Floor
San Jose, California 95113
(408) 938-9060